The Hidden Cost of Functional Approximation Against Careful Data Sizing – A Case Study

Benjamin Barrois
Univ. Rennes 1 – IRISA/INRIA
Email: benjamin.barrois@irisa.fr

Olivier Sentieys
IRISA/INRIA
Email: olivier.sentieys@irisa.fr

Daniel Menard
IETR/INSA, Rennes, France
dmenard@insa-rennes.fr

***Abstract*—Many applications are error-resilient, allowing for the introduction of approximations in the calculations, as long as a certain accuracy target is met. Traditionally, fixed-point arithmetic is used to relax accuracy, by optimizing the bit-width. This arithmetic leads to important benefits in terms of delay, power and area. Lately, several hardware approximate operators were invented, seeking the same performance benefits. However, a fair comparison between the usage of this new class of operators and classical fixed-point arithmetic with careful truncation or rounding, has never been performed. In this paper, we first compare approximate and fixed-point arithmetic operators in terms of power, area and delay, as well as in terms of induced error, using many state-of-the-art metrics and by emphasizing the issue of data sizing. To perform this analysis, we developed a design exploration framework, APXPERF, which guarantees that all operators are compared using the same operating conditions. Moreover, operators are compared in several classical real-life applications leveraging relevant metrics. In this paper, we show that considering a large set of parameters, existing approximate adders and multipliers tend to be dominated by truncated or rounded fixed-point ones. For a given accuracy level and when considering the whole computation data-path, fixed-point operators are several orders of magnitude more accurate while spending less energy to execute the application. A conclusion of this study is that the entropy of careful sizing is always lower than approximate operators, since it require significantly less bits to be processed in the data-path and stored. Approximated data therefore always contain on average a greater amount of costly erroneous, useless information.**

I. Introduction

Throughout its history, computing architectures have shown significant improvements in terms of performance and energy efficiency, mostly thanks to technology evolution forecast by Moore's law. However, in latest semiconductor technologies, energy efficiency is not scaling along with integration capacity since transistor and power budgets are no longer balanced with the end of Dennard's scaling. With the recent enthusiasm for the Internet of Things (IoT) and to reduce power consumption in High Performance Computing (HPC) systems, new ways of still improving energy efficiency must be found.

To face these limitations, research has to focus on computing architectures themselves and, especially on number representations and the way operations are computed. Choosing the right computation precision at the right time during the execution, while preserving the application functionality in reasonable bounds, is another promising approach for improving significantly energy efficiency. In many applications, numbers are represented using floating-point representation, which can reach a high level of accuracy regardless of the data dynamic range. However, this number representation presents an important overhead in terms of performance and energy. This cost can be overridden using fixed-point arithmetic, which represents fractional real numbers as integers. However, this fixed representation is less accurate and has to be managed by the programmer so the application respects a predefined accuracy target. This loss of accuracy is due to the necessity of dropping some bits during calculation, to avoid the use of big operators and to prevent computation from overflows.

This past decade, approximate computing has become a major field of research [1] and a new kind of integer inexact operators emerged [2]. Contrary to accurate, reduced-precision fixed-point operators, the operation result can possibly be false, with a certain quantity of error introduced depending on the inputs. Designing approximate operators explores a new trade-off by intentionally introducing errors to improve performance and energy efficiency, and to overcome limitations of standard high-precision designs. Nevertheless, for a given accuracy target the benefits in terms of energy consumption of these approximate operators has not yet been fairly investigated in comparison to fixed-point arithmetic.

In this paper we propose a fair comparison between fixed-point arithmetic relying on truncated or rounded accurate operators, and inexact operators. These two types of operators are compared in terms of energy consumption, delay, area cost, and computation accuracy with different error metrics. To broaden this comparison, several applications are considered. Finally, to perform this analysis, we developed a design exploration framework, which guarantees that all operators are compared using the same operating conditions.

The rest of the paper is organized as follows. Section II describes fixed-point arithmetic and approximate operators. The framework used for comparison is detailed in Section III. The comparison results for addition and multiplication operators are given in Section IV and for real applications in Section V before concluding in Section VI.

II. Related Work

In this paper, all experiments are based on fixed-point arithmetic, meaning integers are used to represent fractional real numbers. On one side, accurate fixed-point operators are used and the data bit-width is adapted using truncation

or rounding, thus generating computation errors. This bit-width optimization is the only source of inaccuracy. On the other side, approximate operators are used. Their source of inaccuracy is the approximations they perform by design. This section presents the state of the art of fixed-point arithmetic and introduces the approximate operators used in this paper.

A. Fixed-Point Arithmetic

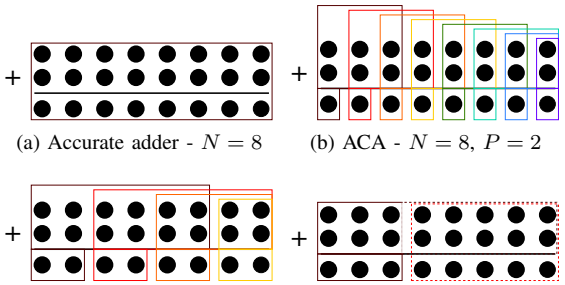
Fixed-point (FxP) arithmetic is widely used in low-power systems. This arithmetic uses basic integer operators and allows exploring the trade-off between accuracy and energy by adjusting the data bit-width. The fixed-point conversion aims at optimizing the data bit-width N to minimize the energy consumption. First the data dynamic range is determined to allocate the minimal number of bits m for the integer parts which guarantees no overflow. Then, the fractional part bit-width n is optimized to provide a minimal computation accuracy. A FxP real number x is approximated as an N-bit signed integer X scaled by a fixed factor, such as $\widehat{x} = X.2^{-n}$. The quantization process corresponding to the elimination of some bits of the n-bit fractional part leads to an unavoidable error. This error can be modelled by a uniformly distributed white noise [3]. The noise power is linked to the number of bits of the fractional part after the quantization process. By considering the classification proposed in [4], quantization noise is a *fail small* error type *i.e.* the error has a high occurrence but with a small amplitude. In this paper, operators whose output is truncated (resp. rounded) will be denoted as OP_t (resp. OP_r).

B. Approximate Hardware Operators

This study is performed using five state-of-the-art approximate operators – three adders and two multipliers. All were chosen for the good balance between their structural simplicity and their acceptable accuracy in comparison to other approximate operators, based on their claims and on literature. The three adders will be referred as ACA (Almost Correct Adder) [5], ETAIV (Error-Tolerant Adder type IV) [6] and RCAApx (Approximate Ripple-Carry-Adder) [7]. The two multipliers will be referred as AAM (Approximate Array Multiplier) [8] and ABM (Approximate Booth Multiplier) [9].

The structure of the three adders is schematized in Figure 1. Each input and output bit of the adders is represented by a black dot. A coloured rectangle on the inputs shows what bits are considered for computing the value of the output bits which are in a rectangle of same colour. A full line represents an accurate sum, while a dotted line represents an approximated addition, used only for RCAApx.

ACA [5] is probably the most classical and referenced inexact adder. ACA can be configured by its number of input bits N and the number of bits for its carry-in prediction P. Therefore, for each of each bit of rank i, the output is obtained by computing the accurate sum of $a_i a_{i-1} \dots a_{i-(P-1)} a_{i-P}$ and $b_i b_{i-1} \dots b_{i-(P-1)} b_{i-P}$. ETAIV [6] is the result of a series of improvements of ETAII [10]. It is configured by N and the number of bits per block X. Indeed, ETAIV is divided into N/X blocks of size X. Each of these blocks performs the accurate sum of $a_{N-kX-1} \dots a_{N-(k+1)X}$ and $b_{N-kX-1} \dots b_{N-(k+1)X}$, taking an input carry generated considering the outputs of the two previous blocks. RCAApx [7] is very different from the two others since it relies on the approximation of the full-adder (FA) logic. It is composed of two parts: the MSB part, which is an accurate adder, and the LSB part composed by the approximate FA. Three types of approximate FA can be used, as defined in [7]. RCAApx can be configured by N, the number of output bits computed using an accurate adder M, and the type of approximate FA used $FAType$ (1, 2 or 3 sorted by decreasing accuracy).



(a) Accurate adder - $N = 8$ (b) ACA - $N = 8$, $P = 2$

(c) ETAIV - $N = 8$, $X = 2$ (d) RCAApx - $N = 8$, $M = 3$

Fig. 1: Structure of the considered approximate adders

The first multiplier to be presented here is AAM [8]. It is based on a fixed-width accurate array multiplier, which means that $N \times N$ input bits result in N output bits instead of $2N$. To obtain AAM, half of the array multiplier basic cells are pruned above its first diagonal (in the classical representation of an array multiplier) and a compensation is performed using a simple series of *AND* and *OR* gates along this diagonal.

ABM [9] is a Radix-2 pruned Modified-Booth (MB) encoding-based multiplier. Radix-2 MB encoding uses one of the input as a coder for the addition grid, allowing a division by 2 of its size. Therefore, the area and delay of the resulting multiplier are improved, despite an overhead for the coder and the decoder which is necessary to convert the output from a redundant to a two's complement representation. However, redundant representation can be advantageously used to perform further calculation, hence the overhead of the decoder can be neglected. ABM approximates the multiplication by pruning half of the summand grid, with a compensation circuit using the most significant bits of the dropped part.

These approximate operators leads to output error compared to the exact value for some input values. These error have a low occurrence but can have a high amplitude. Errors associated with approximate operators are in the *fail rare* and *fail moderate* error type [4].

III. Automated Comparison Framework

In this section, we define the framework developed to compare fixed-point and approximate operators. We also define the error metrics used for accuracy evaluation. The proposed automated framework ApxPerf, depicted in Figure 2, performs a totally automated characterization of a given operator

and uses VHDL and C/C++ operator descriptions. APXPERF framework will be soon delivered as open-source.

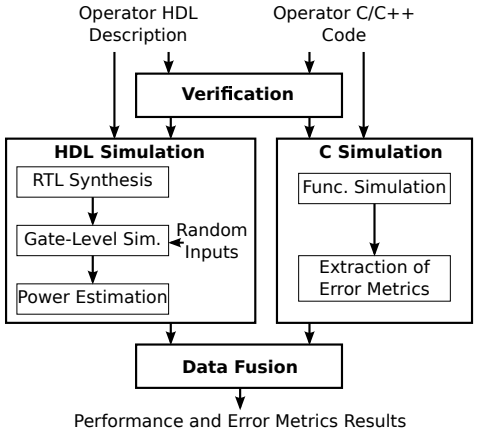


Fig. 2: APXPERF framework

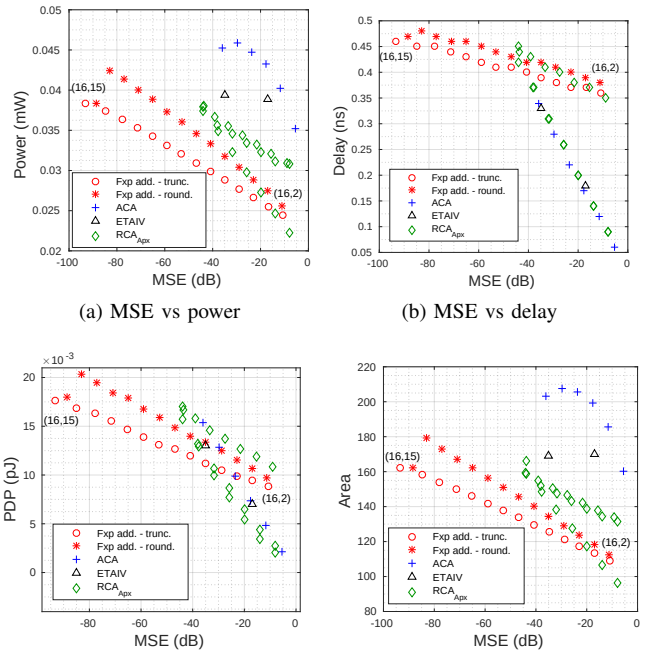
To compare approximate arithmetic operators in a fair way, the operators are all generated and tested using the same constraints and conditions. An arithmetic operator is specified mainly at the register-transfer level (RTL), with some structure details sometimes expressed at the gate level. From this specification, delay constraints, and any technology library, APXPERF generates detailed delay, area, and power results for the operator, using a three-step design flow. First, RTL synthesis is used to obtain an optimized gate-level netlist as well as area and timing information. Second, the gate-level description is simulated using a customizable number of random inputs to generate an activity file. Finally, an analysis is performed using this file and the technology library to estimate power consumption.

In parallel, a functional characterization of the given operator is performed, using its C/C++ description. After an automated validation of the VHDL and C file equivalence, an important number of error metrics are extracted from the operator, using a customizable number of random inputs, leveraging OpenMP simulation acceleration using computing clusters. Then, hardware and functional generated results are gathered and saved as a Matlab MAT file. Matlab functions and scripts are provided to browse and process these results.

Several kinds of error metrics must be considered to characterize operators. Indeed, depending on the application the operator is supposed to be used in, different metrics can be representative of its accuracy. For instance, metrics taking the significance of each bit into consideration can be used in a signal processing application, whereas other metrics would fit better for an application for which any bit position has the same significance, such as most data compression algorithms. The error $e = x - \widehat{x}$ is the difference between the approximate and exact integer outputs. x_k is the k-th bit of the N-bit integer x. In this paper, Mean Square Error (MSE) $E\left[e^2\right]$ and Bit Error Rate (BER) $\frac{1}{N} E\left[\sum_{k=0}^{N} x_k \oplus \widehat{x}_k\right]$ are considered for accuracy metrics. To provide detailed information for each type of application, APXPERF moreover embeds the following metrics: mean error (or error bias) $\mu_e = E[e]$, Mean Average Error (MAE) $E[|e|]$, less penalizing than MSE for high amplitude errors, relative error $E\left[\frac{x-\widehat{x}}{x}\right]$, inferior and superior bound of error $(min_e,\ max_e)$, error rate $P[x - \widehat{x} \neq 0]$, positional BER $E[x_k \oplus \widehat{x}_k]$ for all bit position k, Acceptance Probability (AP) given several values of Minimum Acceptable Accuracy (MAA) [11], Probability Density Function (PDF), and Power Spectral Density (PSD) of error e.

IV. RAW RELATIVE PERFORMANCE ANALYSIS

A first comparison of fixed-point and approximate operators consists in comparing the accuracy of each of them regarding a performance metric (energy, area, delay). For this, all approximate operators of Section II-B were compiled and tested with a number of bits varying from 2 to 32 and all possible combination of parameters. FxP operators (i.e. classical integer adders and multipliers) were tested in the same way, with all possible combinations of input and output size (inputs from 2 to 32, outputs from 2 to 32 for the adders and 2 to 64 for the multipliers). All power results are given for a clock frequency of 100 MHz. The resulting files and useful graphs are generated using APXPERF. Design Compiler (2013.03) was used for RTL synthesis with a 28nm FDSOI technology library, Modelsim (10.3c) for gate-level simulation and PrimeTime (2013.12) for power analysis. Simulation and power analysis are performed on 10^5 random input samples. The extraction of error metrics based on the C description was computed on more than 10^7 random inputs.



(a) MSE vs power

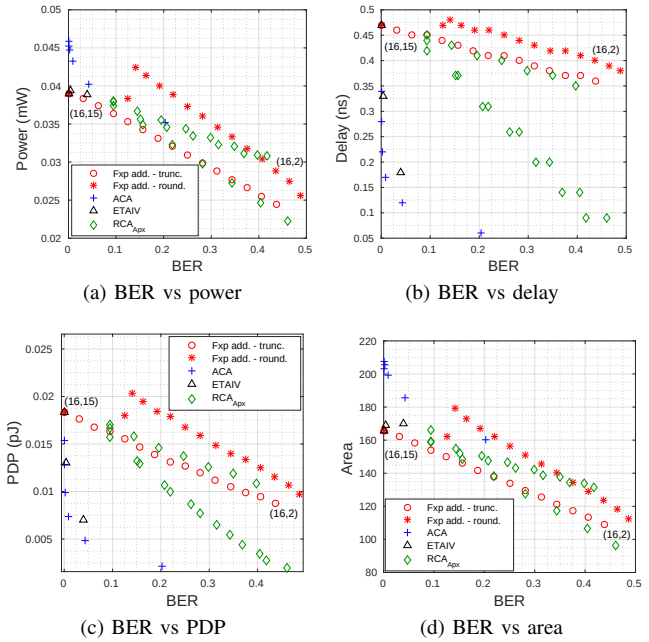
(b) MSE vs delay

(c) MSE vs PDP

(d) MSE vs area

Fig. 3: Direct comparison of 16-bit-input fixed-point and approximate adders regarding MSE

Results for adders are presented on Figure 3 and provide MSE versus power, area, delay, and power-delay product



(a) BER vs power (b) BER vs delay

(c) BER vs PDP (d) BER vs area

Fig. 4: Direct comparison of 16-bit-input fixed-point and approximate adders regarding BER

	$\text{MUL}_t\,(16,16)$	AAM (16)	ABM (16)
Power (mW)	0.273	0.359	0.446
Delay (ns)	0.91	1.23	0.57
PDP (pJ)	0.249	0.442	0.446
Area (μm^2)	805.2	665.5	879.5
MSE (dB)	-89.1	-87.9	-9.63
BER (%)	23.4	27.7	27.9

TABLE I: Direct comparison of 16-bit-input and output fixed-point and approximate multipliers

(PDP). For the sake of clarity, results are here only presented for 16-bit input operators. 16-bit output is considered as the correct adder and used as reference. Truncated and rounded FxP adder outputs vary from 15 to 2 bits (from left to right). For approximate adders, results are given by varying: the number of approximated LSBs (M) and types of FA for RCAApx, the maximal size of carry chain (P) for ACA, and the block size (X) for ETAIV. What can be first noticed is that in terms of power consumption and design area, fixed-point operators are better than approximate operators for a same MSE, except for very-low accuracy. However, in terms of delay, most approximate operators are faster, but they cannot reach the same level of accuracy when more than 8 bits are kept for the fixed-point output. In terms of energy, the PDP of FxP adders is quite near from approximate ones when less than 8 output bits are kept. However, ACA and RCAApx are able to spend less energy without sacrificing much accuracy.

In some applications, all output bits have the same weight on the error. Therefore results on BER metric are presented on Figure 4 for the same adders than previously. Approximate operators achieve very good BER performance when compared to fixed-point operators. Considering the power and area, truncated and RCAApx adders perform similarly for any fixed BER. However, for delay and energy per addition, most approximate operators perform significantly better truncated or rounded FxP operators. When not considering bit significance in the operands, FxP operators are penalized by the suppression of part of their output bits, implicitly forcing them to zero.

Results for the multipliers are presented in Table I. The 16 to 32 integer multiplier is considered as the correct multiplier for accuracy reference. As AAM and ABM multipliers are fixed-width operators (16-bit inputs and output), comparison results are provided only for the truncated FxP multiplier with 16-bit output ($\text{MUL}_t(16,16)$). Fixed-width MUL_t truncated multiplier reaches the best accuracy, and consumes least power. Although MUL_t is slower than ABM, its energy per multiplication (PDP) is 44% better than both approximate operators. ABM is 37% faster than MUL_t and AAM is 17% smaller. However, ABM is extremely MSE inaccurate, with 7 orders of magnitude more erroneous results than fixed-point. Both AAM and ABM are worse than MUL_t by about 19% for BER metric.

As a conclusion for this operator-level performance analysis of various approximation schemes, fixed-point operators perform better when considering the MSE metric representative of signal processing applications, while approximate adders show good BER performance. However, the importance of output bit-width was not taken into account in this results. Indeed, when the bit-width is reduced, as in truncated or rounded operators, the amount of data to transfer to load and store operator inputs and output is consequently reduced. This shortening in bit-width has a major impact on energy consumption and must be considered for real-life application. Thus, although inexact and truncated-or-rounded operators seem to reveal the same gross performance, selecting the second one will allow to decrease energy cost by avoiding the transfer and memory storage of useless erroneous bits.

V. Application-based comparison

In this section, the effect of fixed-point and approximate adders and multipliers is evaluated on different real-life applications, leveraging relevant and adapted metrics. Considered applications include Fast Fourier Transform (FFT), JPEG image encoder, motion compensation filtering in the context of High-Efficiency Video Coding (HEVC) decoding, and K-means clustering.

A. Fast Fourier Transform

As a classical computation kernel used in many signal processing or communication applications, FFT is relevant for this study. ApxPerf provides an instrumented, tunable FFT kernel. This sections presents results on a 32-sample Radix-2 FFT computed on 16-bit input/output data. In a first experiment, only the adders are considered. The total energy to compute the FFT is estimated by

$$PDP_{FFT} = \sum_{i=1}^{N_{add}} PDP_{add,i} + \sum_{i=1}^{N_{mul}} PDP_{mul,i} \qquad (1)$$

where N_{add} and N_{mul} are the total of additions and multiplications, respectively. Figure 5 shows PDP_{FFT} as a function

of output Peak Signal-to-Noise Ratio (PSNR). PSNR is the maximal power of the output signal divided by the MSE, i.e., $PSNR(x)\ [dB] = 10.\log\left[\frac{\max(x^2)}{MSE(x)}\right]$. The exact multipliers used alongside the modified adders are optimally sized according to the adder bit-width, so they are not source of error. For any accuracy constraint, FxP adders (truncation or rounding) notably dominate approximate adders. This supremacy could be explained by two factors: the relative energy cost of multipliers with regards to adders and the need for less operand size for the multiplier when reducing the accuracy of additions. This figure also shows the great potential of energy reduction when playing with accuracy of the fixed-point operators. A first conclusion here is that reducing the FxP adder size provides a smaller entropy of the data processed, transported and stored, than keeping the same bit-width along the computations but containing approximations. The same experiment is performed using 16-bit AAM and

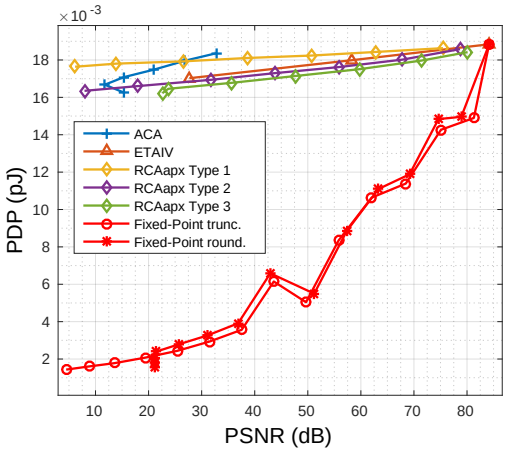


Fig. 5: Power consumption of FFT-32 versus output PSNR using 16-bit approximate adders

ABM multipliers and a 16-bit truncated FxP multiplier, while keeping 16-bit exact adders. Table II shows that AAM and FxP multiplier differ only by 6 dB of of accuracy. However, AAM consumes 78% more energy that fixed-point. Results on the FFT comfort the conclusion of Section IV. Providing results with both approximate adders and multipliers in the same simulation will not lead to a different conclusion.

	$\text{MUL}_t\,(16,16)$	AAM (16)	ABM (16)
PSNR (dB)	53.88	59.66	−18.14
PDP (pJ)	0.249	0.442	0.446

TABLE II: Accuracy and energy consumption of FFT-32 using 16-bit fixed-width multipliers

B. JPEG Encoding

The second application is a JPEG encoder, representative of image processing. The main algorithm of this encoder is the Discrete Cosine Transform (DCT). To obtain an approximate version of the encoder, DCT operations are computed using fixed-point or approximate operators. The quality metric to compare the exact and the approximate versions of the JPEG encoder is the Mean Structural Similarity (MSSIM) [12], which is representative of the human perception of image degradation. This metric results in a score between $[0, 1]$, 1 representing a perfect quality. To obtain Figure 6, the DCT energy consumption is compared for all presented approximate adders, as well as for fixed-point versions. The algorithm is applied with an encoding effort of 90% on the image *Lena*. As observed for the FFT, the fixed-point versions of the algorithm are much more energy efficient than for approximate operators, mostly thanks to the bits dropped during the calculation.

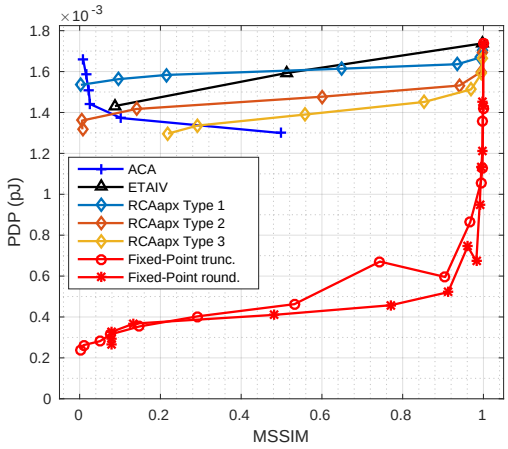


Fig. 6: Power consumption of DCT in JPEG encoding versus output MSSIM using 16-bit approximate adders

C. Motion Compensation filter for HEVC decoder

HEVC is the new generation of video compression standard. Efficient prediction of a block from the others requires fractional position motion compensation (MC) carried-out by interpolation filters. These MC filters are modified using fixed-point and approximate operators to test their accuracy and energy efficiency. Previously described MSSIM metric is used to determine the output accuracy of the filter on a classical signal processing image. Table III gives the energy spent by the MC filter replacing all its additions by adders producing an MSSIM of approximately 0.99. In their 16-bit version, ACA and ETAIV can only reach respectively 0.96 and 0.98. In any case and as discussed above, the multiplier overhead provokes an energy consumption which is 4.6 times superior for the approximate version than for the truncated FxP version. For multipliers replacement, Table III shows that both 16-bit AAM and ABM produce an accuracy similar to fixed-width truncated FxP multiplier. Moreover, replacing multipliers by ABM in the MC filter do not lead to an important energy overhead, which makes it competitive considering that its delay is 37% inferior to $\text{MUL}_t(16,16)$ according to Table I. However, AAM suffers from an important energy overhead.

D. K-means Clustering

The last experiment presented in this section is K-means clustering. Given a bidimensional point cloud, this algorithm classifies them finding centroids and assigning each point to the cluster defined by the nearest centroid. At the core of K-means clustering is distance computation. For the experiment,

	MSSIM	Adder Energy (pJ)	Min. Mult. Energy (pJ)	Total Energy (pJ)
$\text{ADD}_t(16,10)$	99.29%	1.39E−2	4.39E−2	0.898
$\text{ACA}(16,12)$	96.45%	1.54E−2	2.49E−1	4.20
$\text{ETAIV}(16,4)$	98.02%	1.30E−2	2.49E−1	4.17
$\text{RCA}_{\text{Apx}}(16,6,3)$	99.67%	1.00E−2	2.49E−1	4.12

TABLE III: Accuracy and energy consumption of distance computation for HEVC Filter using 16-bit input adders

	MSSIM	Multiplier Energy (pJ)	Min. Adder Energy (pJ)	Total Energy (pJ)
$\text{MUL}_t(16,16)$	99.918%	2.49E−1	1.83E−2	3.77
$\text{AAM}(16)$	99.909%	4.42E−1	1.83E−2	6.48
$\text{ABM}(16)$	99.907%	2.54E−1	1.83E−2	3.85

TABLE IV: Accuracy and energy consumption of distance computation for HEVC using 16-bit input multipliers

5 sets of 5E3 points of data were generated around 10 random points with a Gaussian distribution. The accuracy metric is the success rate, from 0 to 1, representing the proportion of points classified in the correct cluster. Table V presents the success rate and energy spent in distance computation replacing the exact adders by fixed-point and approximate versions. For truncated fixed-point version, the energy spent in multiplication is inherently inferior to approximate, thanks to the reduction of bit-width, leading to a total energy reduction by more than half for a success rate of 99%, and even by nearly 10 for a success rate of about 86%. Table VI shows K-means clustering success rate and energy spent in distance computation performing multiplication using 16-bit input FxP and approximate multipliers. AAM achieves similar accuracy than fixed-width truncated accurate multiplier, with 99% success rate. However, it presents an energy overhead of 75%. ABM achieves very poor performance for K-means, with only 10% success, which is equivalent to prune 12 output bits of a FxP multiplier.

	Success Rate	Adder Energy (pJ)	Min. Mult. Energy (pJ)	Total Energy (pJ)
$\text{ADD}_t(16,11)$	99.14%	1.55E−2	9.36E−2	2.03E−1
$\text{ACA}(16,12)$	99.10%	1.54E−2	2.49E−1	5.13E−1
$\text{ETAIV}(16,4)$	99.43%	1.30E−2	2.49E−1	5.11E−1
$\text{RCA}_{\text{Apx}}(16,6,3)$	99.67%	1.00E−2	2.49E−1	5.08E−1
$\text{ADD}_t(16,8)$	86.00%	1.27E−2	2.40E−2	6.06E−2
$\text{ACA}(16,8)$	86.06%	9.85E−3	2.49E−1	5.08E−1
$\text{ETAIV}(16,2)$	63.25%	7.00E−3	2.49E−1	5.05E−1
$\text{RCA}_{\text{Apx}}(16,10,1)$	87.29%	1.26E−2	2.49E−1	5.11E−1

TABLE V: Accuracy and energy of distance computation for K-means using 16-bit input adders for different success rates

	Success Rate	Multiplier Energy (pJ)	Min. Adder Energy (pJ)	Total Energy (pJ)
$\text{MUL}_t(16,16)$	99.84%	2.49E−1	1.83E−2	5.15E−1
$\text{AAM}(16)$	99.43%	4.42E−1	1.83E−2	9.02E−1
$\text{ABM}(16)$	10.27%	2.54E−1	1.83E−2	5.27E−1
$\text{MUL}_t(16,4)$	10.87%	2.04E−1	1.24E−3	4.09E−1

TABLE VI: Accuracy and energy of distance computation for K-means using 16-bit input multipliers

In spite of the theoretical competitiveness of approximate operators, their advantages are likely to be lost at application level. Indeed, at the difference of fixed-point operators, accuracy reduction is obtained by simplifying the operator structure but not by reducing the operator output bit-width. This reduces the energy of the considered operator, but does not have a positive impact on the other operators, as for fixed-point.

VI. Conclusion

In this paper, two types of hardware approximation were compared: fixed-point arithmetic relying on truncated and rounded accurate operators with careful data sizing, and approximate operators. A direct comparison using the APX-PERF framework showed that both techniques are competitive, depending on the observed error and performance metrics. However, stepping back observing some state-of-the-art applications showed that using approximate operators often leads to an important overhead since they manipulate larger data containing in average more erroneous useless information bits. More generally, it has been shown that comparing the raw performance of operators does not necessarily reflect their performance in a given application context. Hence, a major stake for hardware approximation is now to be considered at a higher level to take more parameters into consideration. An effort must also be made in the research for more efficient approximate multipliers, since they are responsible for the majority of power consumption in computation-intensive applications. However, considering approximate operators in embedded processors to replace or enhance their integer arithmetic unit would still be a good option, since processor data size is fixed and cannot be application specific.

References

[1] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, pp. 8–22, Feb 2016.

[2] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Proc. Design, Automation Test in Europe Conference (DATE)*, pp. 1–6, 2011.

[3] B. Widrow, I. Kollar, and M.-C. Liu, "Statistical theory of quantization," *IEEE Trans. on Instrum. and Measur.*, vol. 45, no. 2, pp. 353–361, 1996.

[4] E. Nogues, D. Menard, and M. Pelcat, "Algorithmic-level approximate computing applied to energy efficient hevc decoding," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[5] A. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Automation and Test in Europe (DATE)*, pp. 1250–1255, 2008.

[6] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *International SoC Design Conference (ISOCC'10)*, pp. 323–327, Nov 2010.

[7] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: Imprecise adders for low-power approximate computing," in *Int. Symp. on Low Power Elect. Design (ISLPED)*, pp. 409–414, 2011.

[8] L.-D. Van, S.-S. Wang, and W.-S. Feng, "Design of the lower error fixed-width multiplier and its application," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 47, no. 10, pp. 1112–1118, 2000.

[9] T.-B. Juang and S.-F. Hsiao, "Low-error carry-free fixed-width multipliers with low-cost compensation circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 6, pp. 299–303, 2005.

[10] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Proc. 12th Int. Symp. on Integrated Circuits (ISIC)*, pp. 69–72, 2009.

[11] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in dsp," *IEEE Trans. on VLSI Syst.*, vol. 18, no. 8, pp. 1–5, 2009.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, pp. 600–612, April 2004.